Strumenti di analisi dati per esperimenti X e gamma nello spazio
An Introdution to FITS (Flexible Image Transport System)

N.Giglietto

INFN-Bari

Introduction

- Essentially universally used in ground-based astronomy and adopted by NASA astronomy missions at least 30 years ago.

ASTRONOMY & ASTROPHYSICS
SUPPLEMENT SERIES

JUNE 1981, PAGE 363

Astron. Astrophys. Suppl. Ser. 44, (1981) 363-370

FITS : A FLEXIBLE IMAGE TRANSPORT SYSTEM

D. C. WELLS (1), E. W. GREISEN (*) (2) and R. H. HARTEN (**)

Kitt Peak National Observatory Tucson, Arizona, U.S.A.
(*) National Radio Astronomy Observatory Charlottesville, Virginia, U.S.A.
(**) The Netherlands Foundation for Radio Astronomy, Postbus 2, 7990 AA Dwingeloo, the Netherlands

Received March 31, accepted September 4, 1980

Summary.- A format for the interchange of astronomical images and other digital arrays on magnetic tape is described. This format provides a simple but powerful mechanism for the unambiguous transmission of n-dimensional, regularly spaced data arrays. It also provides a method for the transmission of a virtually unlimited number of auxiliary parameters that may be associated with the image. The parameters are written in a form which is easily interpreted by both humans and computers. The FITS format has been adopted for the transmission of astronomical image data by several large observatories including the Very Large Array, the Westerbork synthesis telescope, the Kitt Peak Observatory and the Anglo-Australian Observatory.

Key words : data analysis

FITS – Flexible Image Transport System

- Near-universal in astronomy FITS(has been a standard for astronomical images since 1982 and is recognized by the International Astronomical Union
- Extended to handle non-image data
- FITS *binary table* preferred format for transporting large amounts of tabular data
- Only used in astronomy – c++ free libraries and python interfaces

Introduction (2)

- **'Flexible' is true, as the original image (multi-dimensional array) standard has been extended to include tables (ASCII and binary) and more exotic data structures, including variable-length records (similar to ntuples)**
- **The best thing about a standard is generally that it is a standard**
- **FITS is a formal standard and has an oversight committee that in a very deliberate way keeps it current**
- **NASA FITS Support Office [a virtual office] is a good reference**
- **Other refs here: (http://home.fnal.gov/~neilsen/notebook/astroImagingDataReduction/astroImagingDataReduction.html)**

Introduction (3) why use FITS

- **FITS is stable over time and OS platforms (30 years!)**
- **Need to add metadata (comments, tags, version of sw used to produce the image) to some image produced by diffirent missions**
- **Possibility to write data structures in more complex tables (spectra, light curves, ntuple or trees in ROOT)**

Structure of a FITS File

- FITS files consist of one or more 'Header-Data Units'
- The header part of an HDU is ASCII (i.e., human readable without special software) and describes in a defined way the contents of the HDU.
  - For historical reasons the header is always a multiple of 2880 bytes long, i.e., padded to that length
  - As tables are an extension to the original definition, FITS files with tables require a runt main header with no associated data array, to identify them as files that might have tables in succeeding HDUs
- Headers can be quite descriptive, e.g., in a table defining the units of a quantity and the allowed range
- Owing to the adherence to the standard, general FITS readers for images and tables exist. Good and free.

Struttura base di un file FITS

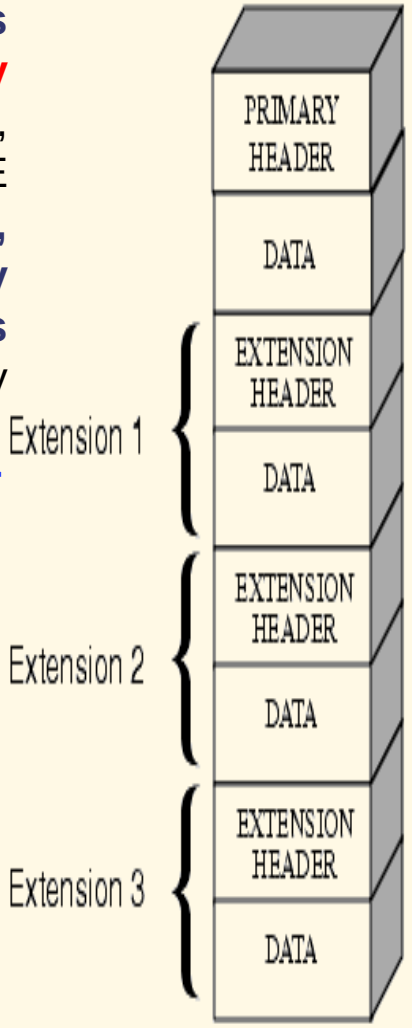
HDUs

A FITS file is comprised of segments called Header/Data Units (HDUs), where the first HDU is called the `Primary HDU', or `Primary Array'. The primary data array can contain a 1-999 dimensional array of 1, 2 or 4 byte integers or 4 or 8 byte floating point numbers using IEEE representations. **A typical primary array could contain a 1-D spectrum, a 2-D image, or a 3-D data cube. Any number of additional HDUs may follow the primary array. These additional HDUs are referred to as FITS `extensions'.** Three types of standard extensions are currently defined:

Image Extensions contain a 0-999 dimensional array of pixels, similar to a primary array (header begins with XTENSION = 'IMAGE ')

ASCII Table Extensions store tabular information with all numeric information stored in ASCII formats. While ASCII tables are generally less efficient than binary tables, they can be made relatively human readable and can store numeric information with essentially arbitrary size and accuracy (e.g., 16 byte reals). (header begins with XTENSION = 'TABLE ')

Binary Table Extensions store tabular information in a binary representation. Each cell in the table can be an array but the dimensionality of the array must be constant within a column. The strict standard supports only one-dimensional arrays, but a convention to support multi-dimensional arrays is widely accepted. (header begins with XTENSION = 'BINTABLE')



FITS World Coordinate System (WCS)

Dettagli http://fits.gsfc.nasa.gov/fits_wcs.html

4 diversi articoli codificano gli standard:

- **Representations of World Coordinates in FITS, Greisen, E. W., and Calabretta, M. R., *Astronomy & Astrophysics*, 395, 1061-1075, 2002**
- **Representations of celestial coordinates in FITS, Calabretta, M. R., and Greisen, E. W., *Astronomy & Astrophysics*, 395, 1077-1122, 2002**
- **Representations of spectral coordinates in FITS, Greisen, E. W., Calabretta, M. R., Valdes, F. G., and Allen, S. L., *Astronomy & Astrophysics*, 446, 747-771, 200**
- **Mapping on the HEALPix grid, Calabretta, M. R. & Roukema, B. F. *MNRAS*, 381, 865, 2007 introduces the HPX spherical projection type as an addendem to the 2nd WCS paper listed above. (see MN.Mazziotta 's talk)**

Header Unit

(da http://fits.gsfc.nasa.gov/fits_primer.html)

Each header unit contains a sequence of fixed-length 80-character keyword records which have the general form:

KEYNAME = value / comment string

The keyword names may be up to 8 characters long and can only contain uppercase letters A to Z, the digits 0 to 9, the hyphen, and the underscore character. The keyword name is (usually) followed by an equals sign and a space character in columns 9 and 10 of the record, followed by the value of the keyword which may be either an integer, a floating point number, a complex value (i.e., a pair of numbers), a character string (enclosed in single quotes), or a Boolean value (the letter T or F). Some keywords, (e.g., COMMENT and HISTORY) are not followed by an equals sign and in that case columns 9 - 80 of the record may contain any string of ASCII text.

Each header unit begins with a series of required keywords that specify the size and format of the following data unit.

A 2-dimensional image primary array header, for example, begins with the following keywords:

SIMPLE = T / file conforms to FITS standard

BITPIX = 16 / number of bits per data pixel

NAXIS = 2 / number of data axes

NAXIS1 = 440 / length of data axis 1

NAXIS2 = 300 / length of data axis 2

The required keywords may be followed by other optional keywords to describe various aspects of the data, such as the date and time of the observation.

COMMENT or HISTORY keywords are also frequently added to further document the contents of the data file.

The last keyword in the header is always the `END' keyword which has blank value and comment fields.

Data Unit

Note that the data unit is not required, so some HDUs only contain the header unit.

The image pixels in a primary array or an image extension may have one of 5 supported data types:

8-bit (unsigned) integer bytes

16-bit (signed) integers

32-bit (signed) integers

32-bit single precision floating point real numbers

64-bit double precision floating point real numbers

The other 2 types of standard extensions, ASCII tables and binary tables, contain tabular information organized into rows and columns. Generally speaking, binary tables are more compact and are faster to read and write than than ASCII tables because the numeric entries are stored in machine readable binary representation.

All the entries within a given column of an ASCII or Binary table extension have the same datatype. The allowed data formats for an ASCII table column are: integer, single or double precision floating point value, or character string. Binary table extensions also support logical (T/F), bit, and complex data formats.

Each entry, or field, in an ASCII table may only contain 1 scalar value. Binary tables are more flexible and allow N-dimensional arrays of of data (either fixed length or variable length) to be stored within each field. Variable-length arrays are implemented by storing a pointer in the field of the table which defines the length and byte offset to the start of the data array which is located in the "heap" area that follows the table proper. The variable-length-arrays convention is not strictly part of the FITS standard but is widely used.

Definizione minima di wcs

Some useful software

- **fv – http://ftools.gsfc.nasa.gov/lheasoft/ftools/fv - interactive editor for FITS files. Also shows images, makes histograms. Many platforms supported, including Windows,MAC Os**
- **FTOOLS futils in general – see http://ftools.gsfc.nasa.gov/lheasoft/ftools - general purpose utilities for manipulating contents and headers of FITS files. For Windows, Cygwin is needed**
- **Image display: DS9 – see http://heawww.harvard.edu/RD/ds9/ - astronomical image display tool, very flexible. A completely functional (as far as I can tell) Windows binary distribution is available**
- **I/O libraries exist for high-level languages and for IDL, Matlab, Python, etc – see http://fits.gsfc.nasa.gov/fits_libraries.html**
- **Python module see http://www.stsci.edu/resources/software_hardware/pyfits**

Evolution: Photoshop+fits plugin (Fits liberator project ESA)

Problem

- **Monochrome, noisy raw FITS data -> clean colour images that represent the Universe:**
  Maximize tonal range, avoid losing resolution, stay honest to data.
- **Surplus of dynamic range in data. Limited dynamic range in output.**
- **Need for speed.**
- **The FITS format is very versatile, and therefore more difficult to 'handle' than e.g. a tiff file.**
- **Science software (IRAF, Midas etc.) limited, slow and inaccessible.**

Solution

- **Photoshop + FITS plug-in.**

The plug-in is free and can be downloaded from:
http://www.spacetelescope.org/projects/fits_liberator

Simple usage: ftools, fv

National Aeronautics and Space Administration
Goddard Space Flight Center
Sciences and Exploration

GO Search HEASARC website
[Advanced Search]
HEASARC Quick Links
---Quick Links---

HEASARC Home | Observatories | Archive | Calibration | Software | Tools | Students/Teachers/Public

NASA's HEAS

FITSIO | FTOOLS | FITS

HEASARC Software

The following software is supported by the HEASARC. Most of this software is designed for professional researchers and advanced students for the analysis of scientific astronomical observations in FITS format. The 'fv' program is also suitable for more general use by amateurs and educators for viewing astronomical image files. These software packages are to be downloaded to the users machine. Web-based tools are available on our tools page.

HEASARC Software Packages

- FITSIO - A subroutine library for reading and writing FITS files for C and Fortran programmers.
- FTOOLS - General FITS file utility programs and mission-specific data analysis tools.
- FV - Interactive editor and viewer for astronomical data files in FITS format. Also provides access to Hera.
- HEAsoft - A unified release of the FTOOLS and XANADU packages.
- Hera - Run the FTOOLS and XANADU software on the Hera servers at the HEASARC, without having to install the software packages locally.
- Maki - Displays fields of view for various instruments on FITS images. Currently supports the Suzaku XIS & HXD, Chandra ACIS & HRC, XMM EPIC, and RXTE PCA detectors.
- PIMMS - Program to estimate count rates from fluxes or vice versa, or to estimate count rates in one instrument from those measured in another.
- Profit - GUI tool for visualizing and modeling high-resolution spectra.
- SAOImage DS9 - Astronomical Imaging and Data Visualization Application: DS9 supports FITS images and binary tables, multiple frame buffers, region manipulation, and many scale algorithms and colormaps.
- SkyView-In-A-Jar - Running *SkyView* on your own computer.
- XANADU - Suite of spectral (xspec), timing (xronos), and image (ximage) analysis programs.
- XSELECT - Multipurpose tool for filtering event files and generating images, spectra, and light curves. Distributed as part of the **HEASARC** package on the download page.
- XSTAR - Program for calculating physical conditions and emission spectra in photoionized gases.

ASTRO-Update - **Latest Versions of Useful Multi-mission Software**

FITS - **Data Formats, Standards and HEASARC Conventions for High-Energy Astrophysics Data**

LAMBDA Toolbox - **Useful Software and Web Tools for Cosmic Microwave Background and General Astronomy**

- **http://hea**

Fv the simplest fits browser

- **Windows like program easy to use and install**
- **→tutrorial → → →**

Lavorare con i file fits - FTOOLS

- **FTOOLS is a collection of utility programs to create, examine, or modify data files in the FITS (Flexible Image Transport System) format.**
- **http://heasarc.nasa.gov/lheasoft/ftools/ftools_subpacks.html**

FITSIO | FTOOLS | FV | HEASoft | Hera | Maki | PIMMS | PROFIT | Xanadu | Xselect | XSTAR | ASTRO-Update | FITS

Sub-packages

For convenience, the individual FTOOLS tasks are collected together in "sub-packages." Thus, the tasks that perform generic functions that could be appropriate for any type of FITS file (e.g., list the contents of a FITS file and write it to another FITS file), are distiguished from the mission-specific tasks that perform calibration or analysis functions on various kinds of high-energy astrophysics data.

General Subpackages

The following General subpackages are currently available:

Package	Description
caltools	General Calibration Tasks
fimage	General FITS Image-Manipulation Tasks
futils	General-Purpose FITS Tasks
heasarc	General Tasks for High Energy Astrophysics (**includes Xselect**)
heatools	Next-generation futils
time	Timing-Specific Tasks
xronos	The HEASARC's general timing analysis package

FTOOLS Subpackages: futils

- http://heasarc.nasa.gov/lheasoft/ftools/futils.html
  - **Fimgcreate:**
    Create a FITS primary array image from an ASCII template file. USAGE fimgcreate bitpix naxes datafile outfile DESCRIPTION This task creates a new FITS primary array image from the data values in the input ASCII template file. If the ascii datafile is given none/NONE, then the task creates an empty image file with pixel value 0.0 EXAMPLES 1. Create a FITS image called 'outfile.fits' containing 200 x 300 short-integer pixels from a data file called 'data.lis' and a header file called 'keywords.lis'. > fimgcreate 16 200,300 data.lis outfile.fits headfile=keywords.lis

FTOOLS Subpackages: futils

- **fcreate:**
  - **fcreate -- Create a FITS table extension from ASCII files describing the format and content of the table.**
  - **USAGE fcreate cdfile datafile outfile**
  - **DESCRIPTION This task creates a new FITS file containing a table extension (either ASCII or binary) by reading two ASCII format files containing (1) a description of the table column formats and (2) the data contents of the table. Optionally, a third ASCII file may be specified (the headfile parameter) which lists additional keywords to be added to the header of the FITS table. Options are provided to skip a specified number of rows at the beginning of the data file and to process only a specified number of rows in the data file. By default all the rows will be processed and written to the FITS table. The data file may be read either in free format (default) or fixed format.**
  - **Example→**

EXAMPLES 1.

- Create a FITS file called 'outtab.fits' with a binary table extension from a column description file called 'coldesc.lis', a free-format data file called 'data.lis' and a header file called 'keywords.lis'.
- **fcreate coldesc.lis data.lis outtab.fits headfile=keywords.lis**
- The column description file (coldesc.lis) could contain: name 12A
  ra E degrees
  dec E degrees
  rank B
  The data file (data.lis) could contain:
  'Alpha Boot' 15.345 -12.346 3
  Vega 123.225 39.245 7
  'RR Lyrae' 34.46788 -36.9 1
  The optional header file (keywords.lis) could contain:
  INSTRUME ROSAT / name of instrument
  OBS-DATE 23/08/92 date of the observation
  observer 'I. Newton'
  principle investigator
  tempdet 98.6 / temperature of the detector
  history This file was created with the ftools fcreate task

Altre funzioni di utilità

- Faddcol per aggiungere una colonna ad un file esistente:
  - Es:1. faddcol target.fits col.fits time,x,y
    The columns named time, x, and y will be copied from the first extension in the file col.fits to the first extension in the file target.fits.
  - Es2. faddcol target.fits[3] col.fits -time
    All the column in the first extension in the file col.fits, except the 'time' column will be copied to the 3rd extension in the file target.fits.
- Fdelcol rimuove una colonna da un file esistente

Operazioni con colonne

fcalc -- Calculates values for a column using an arithmetic expression.

USAGE fcalc infile[ext#] outfile clname expr

(see http://heasarc.nasa.gov/lheasoft/ftools/fhelp/fcalc.txt for details)

EXAMPLES

1. **fcalc input.fits area.fits AREA "X*Y“** (calcola il prodotto ed il risultato è riportato come colonna AREA)
2. **fcalc twovec.fits dot.fits DOT "A[1]*B[1] + A[2]*B[2]" (prodotto scalare)**
3. Normalize a spectrum in the vector column SPEC by its average value:
   fcalc spec.fits out.fits SPEC "SPEC/(SUM(SPEC) / NELEM(SPEC))"
4. Count the number of pixels in a scanline, held in vector column SCAN, with intensities above a value given by a detection-threshhold keyword, DETECT, putting the result in a new column CNT:
   fcalc scan.fits out.fits CNT "SUM(SCAN>DETECT)“2

Ftcalc (heatool utilities)

http://heasarc.nasa.gov/lheasoft/ftools/headas/heatools.html

ftcalc infile[ext] outfile column expression

- Mathematical operators: +, -, *, /, ** or ^ (exponentiation)
- Boolean operators in C or Fortran-type notation: .eq., ==, .ne., !=, .lt., <, .le., <=, .gt., >, .ge., >=, .or., ||, .and., &&, .not., !, and ~ (approximately equal, to within 1E-07)
- **Math library functions: abs(x), cos(x), sin(x), tan(x), arccos(x), arcsin(x), arctan(x), arctan2(y,x), cosh(x), sinh(x), tanh(x), round(x), floor(x), ceil(x) exp(x), sqrt(x), log(x), log10(x), x%y (modulus), random() (returns random number >= 0 and < 1), randomn() (returns Gaussian distribution with zero mean and unit standard deviation), randomp(x) (returns a Poisson random distribution whose expected number of counts is X. X may be any positive real number of expected counts, including fractional values, but the return value is an integer.) min(x,y), max(x,y), accum(x), seqdiff(x), angsep(ra1, dec1, ra2, dec2) (all in degrees).**

String constants: enclose string values in quotes, e.g., 'Crab', 'M101'

Datatype casts to convert reals to integers or integers to reals: (int) x, (float) i

Conditional expressions: 'b?x:y' where expression 'x' is evaluated if 'b' is TRUE (not equal to zero), otherwise expression 'y' is evaluated.

Test for near equality: near(value1, value2, tolerance) returns 0 if value1 and value2 differ by more than tolerance.

Bit masks: The 'x' character represents a wild card: b11x001 (binary), o447x1 (octal), h0FxD (hex)

EXAMPLES:

- **Compute a 3 point running mean of the values in the 'Rate' column and write result to a single precision floating point column called 'Smooth'. A double precision column would have been created by default if the 'tform' parameter had not been specified.**
  ftcalc input.fits outfile.fits Smooth '(Rate{-1}+Rate+Rate{1})/3.' tform='1E'
- **First, create a virtual input file that contains only the X and Y columns in the table, then compute the distance between a point whose X and Y coordinates are defined by the XCENTER and YCENTER keywords, and the positions defined by the X and Y column values in each row of the table. Store the result in a column called 'Radius'.**
  ftcalc 'input.fits[col X;Y]' outfile.fits Radius 'sqrt((X - #XCENTER)2 + (Y - #YCENTER)**2)'**
- **Conditionally compute the 'Rate' column. If 'Counts' is greater than zero then Rate is calculated by dividing the 'Counts' column value by the 'EXPOSURE' keyword value. If the 'Counts' value is less than or equal to zero, then the Rate value is set = 0.**
  ftcalc input.fits outfile.fits Rate 'Counts>0?Counts/#exposure:0.'

Lavorare con i file FITS: pyfits x python

- **Con gli ftools e le futils è virtualmente possibile fare ogni operazione di manipolazione sui file fits ed una loro diretta visualizzazione. Spesso pero' abbiamo da leggere dati da varie fonti, vogliamo produrre istogrammi di variabili in vari file o semplicemente vogliamo utilizzare altri standard di uscita (ROOT ad esempio). In casi del genere python è un linguaggio che agevolmente permette di condividere e trattare dati e sw eterogenei con un linguaggio di programmazione semplice ed intuitivo. Vi è una libreria di interfaccia a FITS chiamata pyfits che permette di accedere ai dati contenuti nei file FITS:**
- **http://www.stsci.edu/resources/software_hardware/pyfits**
- **Una breve introduzione a python è qui http://giglietto.ba.infn.it/%7egiglietto/corso_python_print.pdf**

Pyfits tutorial

Assuming pyfits has been installed on your laptop/workstation the basic instruction to open an existing fits file is

```
>import pyfits
>hdu = pyfits.open('input.fits')
>hdulist.info()
..output
Filename: inputs.fits
No. Name Type Cards Dimensions Format
0 PRIMARY PrimaryHDU 220 () int16
1 SCI ImageHDU 61 (800, 800) float32
2 SCI ImageHDU 61 (800, 800) float32
3 SCI ImageHDU 61 (800, 800) float32
4 SCI ImageHDU 61 (800, 800) float32
---
>hdu.close() chiude il file
```

→→TUTORIAL→→

Esempio di python script1

MACRO 1 (simple.py) uso: ./simple.py nomefile.fits

```
#!/usr/bin/env python <-- dichiarazione che si fa in tutte le shell eseguibili
import math
import os
import sys,string
import pyfits <= importa pyfits
tag =""
if sys.argv[1:]: <= controlla se c'e' un argomento come input
fname = sys.argv[1] <= si aspetta come argometo il nome del file
print fname
# trova se nel filename c'e' la parola mn ad esempio
if fname.find('mn') >-1:
tag="MOON"
print tag
# apre il filename con pyfits vedere il manuale per altri comandi

hdulist=pyfits.open(fname)
print "ok"
table=hdulist[1].data <= nei file fits il secondo blocco contiene i dati
(numerazione alla c++ parte da 0) campo è tipo dati
<il primo blocco è l'header e contiene varie informazioni>
```

→

```
time=table.field('TIME') <= per esempio prendo i tempi
print time
tmin=min(table.field('TIME'))
tmax=max(table.field('TIME'))
print "TMIN TMAX=",tmin,tmax

zen=table.field('ZENITH_ANGLE')
ra=table.field('RA')
dec=table.field('DEC')
srcid=table.field('MC_SRC_ID')

print "NUMERO DATI ",ra.size()
#esempio di loop
for i in range(ra.size()):
# print ra[i]
var=time[i]
if 2.51e8< var <2.53e8 :
# print var
print "Src ID: ",srcid[i]
hdulist.close()
```

macro2 interfacciata a root (simple_root.py)

```
#!/usr/bin/env python
import math
import os
import sys,string
import pyfits
import ROOT

from ROOT import TCanvas, TH2F, TH1F, TPad, TFile, TPaveLabel, TPaveText
from ROOT import gROOT



gROOT.Reset()

c1 = TCanvas( 'c1', 'Histogram Drawing Options', 200, 10, 700, 900 )
#h1f = TH1F( 'h1f', 'time', 200, 0, 10 )
tag =""
if sys.argv[1:]:
fname = sys.argv[1]

hdulist=pyfits.open(fname)
table=hdulist[1].data
time=table.field('TIME')
tmin=min(table.field('TIME'))
tmax=max(table.field('TIME'))
print tmin,tmax
```

```
zen=table.field('ZENITH_ANGLE')
ra=table.field('RA')
dec=table.field('DEC')

myfile = TFile( tag+'out.root', 'RECREATE' )

h1f = TH1F( 'h1f', 'time', 1000, tmin, tmax )

trate = TH1F( 'trate', 'events vs time from solarObj', 2000, tmin, tmax )
radec = TH2F('radec',' radec ', 100,0., 360.,100,-90.,90.)
print ra.size()
for i in range(ra.size()):
    var=time[i]
    radec.Fill(ra[i],dec[i])
    h1f.Fill(var)
h1f.Draw()
c1.Update()
c1.Print("time.ps")
myfile.cd()
myfile.Write()
myfile.Close()
```

Esempio più comune cercare da un catalogo astronomico

- http://heasarc.nasa.gov/cgi-bin/W3Browse/w3browse.pl
- **Oppure**
- http://fermi.gsfc.nasa.gov/cgi-bin/ssc/LAT/LATDataQuery.cgi

- **Si parte da questo motore per recuperare dati o catalogo**
- **Scopo: cercare in una regione di cielo qualche oggetto associato ad una mia osservazione**

Esempio usare pyfits per selezionare da un catalogo

```
>>> import pyfits
>>> ftcat="gll_pscP72Y_v5r2_flags_assoc_v5r11p3.fit"
>>> hdu=pyfits.open(ftcat)
>>> hdu.info()
Filename: gll_pscP72Y_v5r2_flags_assoc_v5r11p3.fit
No.  Name        Type      Cards  Dimensions  Format
0    PRIMARY     PrimaryHDU      4  ()          uint8
1    LAT_POINT_SOURCE_CATALOG  BinTableHDU    294  1873R x 66C
     [18A, 18A, E, E, E, E, E, I, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E,
     E, E, E, 18A, E, E, E, E, L, E, E, E, E, E, E, E, E, E, E, E, 5E, 5E, E, E, E, E,
     E, E, 24E, 24E, 24E, 24E, I, I, 325A25, 13E, 13E, 13E, 13E, 13I]
2    ID_CAT_REFERENCE  BinTableHDU    17  36R x 4C      [I, 50A, 255A,
     255A]
3    EXTENDEDSOURCES  BinTableHDU    60  12R x 11C     [18A, 18A, E, E,
     E, E, 24A, E, E, E, 68A]
4    ROIS        BinTableHDU    136  933R x 13C    [I, E, E, E, E, E, E, E, E, E, E,
     E, E]
5    ENERGYBOUNDS  BinTableHDU    43  5R x 3C       [E, E, E]
6    HIST_START  BinTableHDU    38  25R x 1C      [D]
7    GTI         BinTableHDU    84  11501R x 2C   [D, D]
```

```
>>> head=hdu[0].header
>>> head.ascardlist()
SIMPLE  =                    T / conforms to FITS standard
    BITPIX  =                    8 / array data type                                NAXIS
    =                    0 / number of array dimensions
    EXTEND  =                    T
>>> head.items()
[('SIMPLE', True), ('BITPIX', 8), ('NAXIS', 0), ('EXTEND', True)]
>>> head['BITPIX']
8
>>> head['NAXIS']
0
```

Ripetiamo l'operazione sulla tabella successiva

```
>>> head1=hdu[1].header
>>> dir(head1)  ← esplora tutte le funzioni della classe
['__contains__', '__delitem__', '__doc__', '__getitem__', '__init__',
    '__iter__', '__module__', '__setitem__', '__str__', '_add_commentary',
    '_hdutype', '_mod', '_strip', '_updateHDUtype', 'add_blank',
    'add_comment', 'add_history', 'ascard', 'ascardlist', 'copy',
    'fromTxtFile', 'get', 'get_comment', 'get_history', 'has_key', 'items',
    'keys', 'rename_key', 'toTxtFile', 'update']
```

```
print head1.ascardlist()
XTENSION= 'BINTABLE'           / binary table extension
BITPIX  =                    8 / array data type
NAXIS   =                    2 / number of array dimensions
NAXIS1  =                 1232 / length of dimension 1
NAXIS2  =                 1873 / length of dimension 2
PCOUNT  =                    0 / number of group parameters
GCOUNT  =                    1 / number of groups
TFIELDS =                   66 / number of table fields
TTYPE1  = 'Source_Name'
TFORM1  = '18A     '
TTYPE2  = 'NickName'
TFORM2  = '18A     '
TTYPE3  = 'RAJ2000 '
……
TTYPE65 = 'ID_Angsep'
TFORM65 = '13E     '
TTYPE66 = 'ID_Catalog'
TFORM66 = '13I     '
TBUCD1  = 'meta.id;meta.main'  / UCD for Source_Name
HDUCLAS1= 'SRCLIST '           / an OGIP standard class
TBUCD3  = 'pos.eq.ra;meta.main' / UCD for RAJ2000
TLMAX3  =                360.0 / Maximum value for RAJ2000
TLMIN3  =                  0.0 / Minimum value for RAJ2000
EXTNAME = 'LAT_POINT_SOURCE_CATALOG' / name of this binary table extension
HDUDOC  = 'GLAST Project Science Data products format document' / Document descr
HDUCLASS= 'OGIP    '           / format conforms to OGIP standard
HDUVERS = '1.0.0   '           / version of the format
CDS-NAME= '2FGL    '           / Catalog Name
STVERS  = 'ScienceTools-RELEASE-09-23-01' / ScienceTools version
PIPVERS = 'Gpipeline-v2r12p15T14' / Pipeline code version
RUNNAME = 'P72Y_uw23'          / Name of Run
RSPFUNC = 'P7SOURCE_V6'        / Response function
TSMIN   =                   25 / Test Statistic Treshold
EMIN    =                100.0 / low limit for the energy band (MeV)
EMAX    =               100000.0 / high limit for the energy band (MeV)
SYSREL  =                  0.0 / Relative systematic error for flux
ERRUPLIM=                  0.5 / Relative error above which the errors are compu
TSUPLIM =                 10.0 / Ts below which the errors are computed from Upp
ERPOSABS=                0.005 / Systematic position error radius
```

Accedere ai dati

```
>>> data=hdu[1].data
>>> dir(data)
```

['T', '__abs__', '__add__', '__and__', '__array__', '__array_finalize__', '__array_interface__', '__array_prepare__', '__array_priority__', '__array_struct__', '__array_wrap__', '__class__', '__contains__', '__copy__', '__deepcopy__', '__delattr__', '__delitem__', '__delslice__', '__dict__', '__div__', '__divmod__', '__doc__', '__eq__', '__float__', '__floordiv__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getslice__', '__gt__', '__hash__', '__hex__', '__iadd__', '__iand__', '__idiv__', '__ifloordiv__', '__ilshift__', '__imod__', '__imul__', '__index__', '__init__', '__int__', '__invert__', '__ior__', '__ipow__', '__irshift__', '__isub__', '__iter__', '__itruediv__', '__ixor__', '__le__', '__len__', '__long__', '__lshift__', '__lt__', '__mod__', '__module__', '__mul__', '__ne__', '__neg__', '__new__', '__nonzero__', '__oct__', '__or__', '__pos__', '__pow__', '__radd__', '__rand__', '__rdiv__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__', '__rfloordiv__', '__rlshift__', '__rmod__', '__rmul__', '__ror__', '__rpow__', '__rrshift__', '__rshift__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr__', '__setitem__', '__setslice__', '__setstate__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__truediv__', '__xor__', '_clone', '_coldefs', '_convert', '_file', '_gap', '_get_scale_factors', '_heapoffset', '_names', '_nfields', '_scale_back', 'all', 'any', 'argmax', 'argmin', 'argsort', 'astype', 'base', 'byteswap', 'choose', 'clip', 'compress', 'conj', 'conjugate', 'copy', 'ctypes', 'cumprod', 'cumsum', 'data', 'diagonal', 'dtype', 'dump', 'dumps', **'field'**, 'fill', 'flags', 'flat', 'flatten', 'formats', **'getfield'**, 'imag', 'item', 'itemset', 'itemsize', 'max', 'mean', 'min', 'names', 'nbytes', 'ndim', 'newbyteorder', 'nonzero', 'prod', 'ptp', 'put', 'ravel', 'real', 'repeat', 'reshape', 'resize', 'round', 'searchsorted', 'setfield', 'setflags', 'shape', 'size', 'sort', 'squeeze', 'std', 'strides', 'sum', 'swapaxes', 'take', 'tofile', 'tolist', 'tostring', 'trace', 'transpose', 'var', 'view']

```
>>> data.names
```

['Source_Name', 'NickName', 'RAJ2000', 'DEJ2000', 'GLON', 'GLAT', 'ROI_dist', 'ROI_num', 'Conf_95_SemiMajor', 'Conf_95_SemiMinor', 'Conf_95_PosAng', 'Test_Statistic', 'Signif_Avg', 'Npred', 'Pivot_Energy', 'Flux_Density', 'Unc_Flux_Density', 'Flux100', 'Unc_Flux100', 'Flux1000', 'Unc_Flux1000', 'Energy_Flux100', 'Unc_Energy_Flux100', 'Energy_Flux1000', 'Unc_Energy_Flux1000', 'Spectral_Index', 'Unc_Spectral_Index', 'PowerLaw_Index', 'TSCurve', 'SpectrumType', 'beta', 'Unc_beta', 'Cutoff', 'Unc_Cutoff', 'Extended', 'Flux100_300', 'Unc_Flux100_300', 'Flux300_1000', 'Unc_Flux300_1000', 'Flux1000_3000', 'Unc_Flux1000_3000', 'Flux3000_10000', 'Unc_Flux3000_10000', 'Flux10000_100000', 'Unc_Flux10000_100000', 'Spectral_Fit_Quality', 'Npred_Bands', 'TS_Bands', 'Variability_Index', 'Signif_Peak', 'Flux_Peak', 'Unc_Flux_Peak', 'Time_Peak', 'Peak_Interval', 'Flux_History', 'Unc_Flux_History', 'TSVar', 'TS_History', 'Flags', 'ID_Number', 'ID_Name', 'ID_Probability', 'ID_RA', 'ID_DEC', 'ID_Angsep', 'ID_Catalog']

```
>>> sources=data.field('Source_Name')
>>> print sources[0:10]
```

['2FGL J0000.9-0748' '2FGL J0001.7-4159' '2FGL J0002.7+6220'
'2FGL J0004.2+2208' '2FGL J0004.7-4736' '2FGL J0006.1+3821'
'2FGL J0007.0+7303' '2FGL J0007.7+6825' '2FGL J0007.8+4713'
'2FGL J0008.7-2344']

```
... >>> id=data.field('ID_Name')
>>> sources=data.field('Source_Name')
>>> id=data.field('ID_Name')
>>> ra=data.field('RAJ2000')
>>> dec=data.field('DEJ2000')
>>> flux=data.field('Flux100')
>>> for i in range(5):
...     print sources[i],id[i],ra[i],dec[i],flux[i]

2FGL J0000.9-0748 BZBJ0001-0746 J000117-074633 J000118-074626 1FGL J0000.9-0745 0.233711 -7.8155 1.27181e-08
2FGL J0001.7-4159 1FGL J0001.9-4158 0.438849 -41.9965 6.54946e-09
2FGL J0002.7+6220 1FGL J0003.1+6227 0.679812 62.3396 3.62804e-08
2FGL J0004.2+2208 1FGL J0004.3+2207 1.05569 22.1365 1.35459e-08
2FGL J0004.7-4736 BZQJ0004-4736 J0004-4736 J0004-4736 J000435-473619 J000436-473610 PKS 0002-478 1FGL J0004.7-4737 1.18021 -47.6116 2.66421e-08
```

Aplpy plot in python using fits

- **http://aplpy.github.com/**

Lavorare con FITS: C++ modules FITSIO

- ftp://heasarc.gsfc.nasa.gov/software/fitsio/c/cfitsio3280.tar.gz

```
-------------------------------------------------------------
      #include <string.h>
      #include <stdio.h>
1:    #include "fitsio.h"
      int main(int argc, char *argv[])
      {
2:              fitsfile *fptr;
                char card[FLEN_CARD];
 3:             int status = 0, nkeys, ii; /* MUST initialize status */
 4:             fits_open_file(&fptr, argv[1], READONLY, &status);
                fits_get_hdrspace(fptr, &nkeys, NULL, &status);
                for (ii = 1; ii <= nkeys; ii++) {
                     fits_read_record(fptr, ii, card, &status); /* read keyword */
                     printf("%s\n", card);
                }
       printf("END\n\n"); /* terminate listing with END */
      fits_close_file(fptr, &status); if (status) /* print any error messages */
5:    fits_report_error(stderr, status);
      return(status);
}
------------------------------------------------------------- This program opens the specified FITS file and
      prints out all the header keywords in the current HDU.
```

JavaFits

- http://heasarc.gsfc.nasa.gov/docs/heasarc/fits/java/v0.9/

Interfaccia con R

- R è un tool di statistica molto potente, al suo interno vi è anche un interfaccia a fits
- http://cran.r-project.org/

Uso dal''interno di R

> install.packages("FITSio") downloads the FITS reader from CRAN and installs it. You need to load the package before using it:

> require("FITSio") Loading required package: FITSio [1] TRUE To look at a header, and head a value for a specific keyword:

> im <- readFITS("http://das.sdss.org/www/cgi-bin/drC?RUN=3630&RERUN=40&CAMCOL=3&FIELD=83&FILTER=r") > im$hdr[1:10] [1] "SIMPLE" "T" "BITPIX" "16" "NAXIS" "2" "NAXIS1" "2048" [9] "NAXIS2" "1489" > im$hdr[which(im$hdr=="AIRMASS")+1] [1] "1.23555004241777" Like pyFITS, R can take a URL in place of a file name.

To read a table and make some plots and histograms:

> require(FITSio) Loading required package: FITSio [1] TRUE > catalog <- readFrameFromFITS("http://das.sdss.org/pt/objects/v8_0v3_4-0/GCL/NGC5053short/kaCalObj-00138142.fit") > attach(catalog) > plot(ra,dec) > hist(mag.2,xlim=c(10,20),ylim=c(0,50),breaks=500) (The handling table cells that are themselves arrays is a little awkward in R.)

You can even display an image in R:

> im <- readFITS("http://das.sdss.org/www/cgi-bin/drC?RUN=3630&RERUN=40&CAMCOL=3&FIELD=83&FILTER=r") > image(im$imDat,zlim=c(1100,1200),col=gray.colors(256))

Approfondimento python1 -scipy

http://www.scipy.org/

SciPy

Download | Getting Started | Documentation | Report Bugs | Read the Blog

Scientific Tools for Python

SciPy (pronounced "Sigh Pie") is open-source software for mathematics, science, and engineering. It is also the name of a very popular conference on scientific programming with Python. The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The SciPy library is built to work with NumPy arrays, and provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization. Together, they run on all popular operating systems, are quick to install, and are free of charge. NumPy and SciPy are easy to use, but powerful enough to be depended upon by some of the world's leading scientists and engineers. If you need to manipulate numbers on a computer and display or publish the results, give SciPy a try!

SciPy is a community effort. We seek volunteers at all levels of ability to work on the project, from coding and packaging to documentation, tutorials, recipes, and the web site. Visit the Developer Zone if you are interested in helping out (or if you have bug reports).

Python and Scientific Computing

NumPy and SciPy are two of many open-source packages for scientific computing that use the Python programming language. This website, together with other subdomains of the scipy.org domain, serves as a

scipy

- **http://docs.scipy.org/doc/**

astropy

- http://www.astro.washington.edu/users/rowen/AstroPy.html
- http://astropy.wikispaces.com/
- http://astropy.org/

https://github.com/astropy/astropy/zipball/master

Coordinate astronomiche di oggetti http://rhodesmill.org/pyephem/

Conversioni coordinate

Esempio di calcolo coordinate astronomiche (pyephem)

```
import ephem
m = ephem.Mars('1970')
print m
dir(m)
m.compute('1986/2/9', epoch='1950')
moon = ephem.Moon()
moon.compute('2011/11/24', epoch='2000')
print moon.size
print moon.size/3600.
moon.ra
dir(moon.ra)
moon.ra.real
moon.dec
moon.dec.real
ephem.degrees(moon.dec)
ephem.degrees(moon.ra)
moon.ra
```

http://rhodesmill.org/pyephem/quick.html

Conclusioni

- **Le analisi dati astro-particellari hanno la necessità di interfacciarsi o di essere presentate ad una comunità astrofisica che tipicamente usa sw e tools di visualizzazione differenti dalla comunità particellare**
- **Per le immaggini lo standard FITS è uno dei più usati ed ha il vantaggio della stabilità nel tempo e facilità di trasporto**
- **Per il sw in generale python è estremamente comodo perché permette l'accesso ad una vasta libreria di codici astronomiche, statistiche in particolare, ed è facile ed intuitivo come linguaggio di programmazione**
- **In generale per la comunità astrofisica è necessario l'accesso a cataloghi di sorgenti con l'accesso a informazioni quali spettri e immagini, il formato fits permette lo scambio tra i diversi strumenti e osservatori**